Evolutionary Model
based on Fisher's Geometric Model

Daniel Duda, Bartosz Skowronek

May 6, 2024

1 Introduction

The goal of this project was to create an evolutionary model based on Fisher's Geometric Model. Subsequently, we have performed a series of simulations to study the effects of mutation probability and effect on the adaptation of the population. Additionally, our specific aim was to implement and evaluate an additional mechanism of interaction between two populations: prey and predator.

2 Methodology

The evolutionary model is based on the following components:

1. Individual:
   - Each individual is represented by a numpy array of shape `(n_traits,)`, where `n_traits` is the number of genetic traits. Each trait is a float number in the range $[-1, 1]$.
   - The number of traits can be easily changed in the code using the `num_genes` parameter.
   - Using a numpy array instead of a list allows for vectorized operations, which is computationally more efficient.
   - The model assumes that all individuals are hermaphrodites, but only one individual can reproduce in each generation.
2. Population:
   - Initially, the population is represented by a list of individuals of length `init_population`, and each individual has genetic traits randomly sampled from a uniform distribution in the range $[-1, 1]$ for each trait.
   - The maximum size of the population is defined by the `max_population` parameter. When the population size reaches the maximum size, the individuals with the lowest fitness values are removed from the population until the population size is equal to `max_population`.
3. Environment:
   - The environment is implemented explicitly in the `Environment` class, which manages multiple populations and simulates their interactions under various scenarios like global warming or meteor impacts.
   - The environment initializes and contains multiple populations, facilitating their interactions and evolution over time.
   - The environment can apply changes to the optimal genotypes of populations, simulating environmental pressures or catastrophic events.
4. Fitness Evaluation:

- The fitness of each individual is calculated using a Gaussian function based on the Euclidean distance between the individual's genotype o and the optimal genotype α: $\phi_\alpha(o) = \exp\left(-\frac{\|o-\alpha\|^2}{2\sigma^2}\right)$ where σ is the fitness coefficient that controls the width of the Gaussian function.
- The Euclidean distance between two genotypes o and α in an n-dimensional space is calculated as: $\|o - \alpha\| = \sqrt{\sum_{i=1}^{n}(o_i - \alpha_i)^2}$ where o_i and α_i are the i-th components of the genotypes o and α, respectively.
- The fitness function is implemented as a vectorized function (`_evaluate_fitness`) and applied to the entire population at once, which is computationally efficient.
- The fitness value is in the range $[0, 1]$ and provides the probability of survival for the individual in the next generation.

5. Mutation:

- Mutation is implemented in the `mutate` method of the `Population` class.
- For each individual, a random number is generated from a uniform distribution. If this random number is less than `mutation_probability`, the individual's traits are mutated.
- For each gene of the individual, a random number is sampled from a normal distribution with mean 0 and standard deviation `mutation_effect`. This value is then added to the corresponding gene, resulting in a new trait value for the individual.
- This process enables the exploration of the search space and introduces genetic diversity into the population.

6. Parent Selection:

- Parent selection is performed using the `select` method of the `Population` class.
- The selection probability for each individual is calculated based on the population size, maximum population size, and the individual's fitness value.
- Individuals are selected as parents if a random number generated from a uniform distribution is less than their selection probability.
- This selection method favors individuals with higher fitness values, but it also allows individuals with lower fitness values to have a chance of being selected as parents, promoting diversity in the population.
- The selected parents are then used to create offspring in the reproduction process.

7. Reproduction:

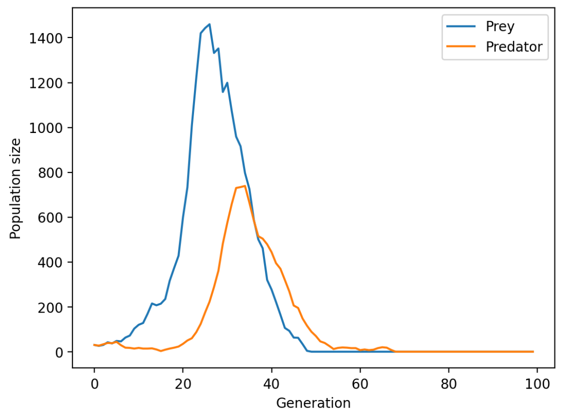
- The reproduction process is implemented in the `reproduce` method of the `Population` class.
- The selected parents are randomly paired up to create offspring.
- For each pair of parents, the number of offspring is determined by sampling from a Poisson distribution with mean `max_num_children`.
- The genotypes of the offspring are created by randomly selecting genes from either parent for each gene position.
- The offspring are then added to the population, and the population size is limited to `max_population` by removing the least fit individuals if necessary.

8. Interaction Between Populations:

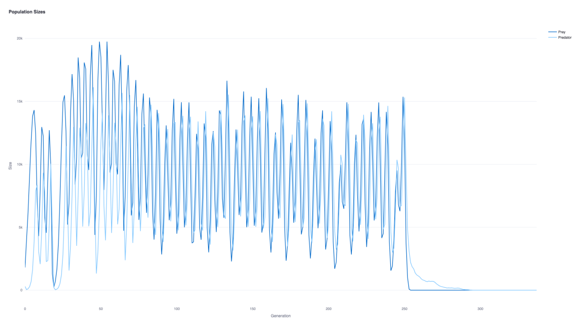
- The interaction between populations (prey and predator) is implemented in the `evaluate` method of the `Population` class.

- The fitness of each individual in a population is adjusted based on the mean fitness and size of the other population.
- The interaction fitness is calculated as: $\phi_{\text{interaction}} = \frac{\bar{\phi}_{\text{other}} \cdot I \cdot f_{\text{other}}}{f_{\text{self}}}$ where $\bar{\phi}_{\text{other}}$ is the mean fitness of the other population, I is the interaction value, f_{other} is the frequency of the other population, and f_{self} is the frequency of the current population.
- The total fitness of an individual is the sum of their genotypic fitness (calculated using the fitness function) and the interaction fitness, clipped to the range $[0, 1]$.
- This interaction mechanism allows the fitness of one population to be influenced by the state of the other population, simulating the dynamics between prey and predator populations.

3 Results

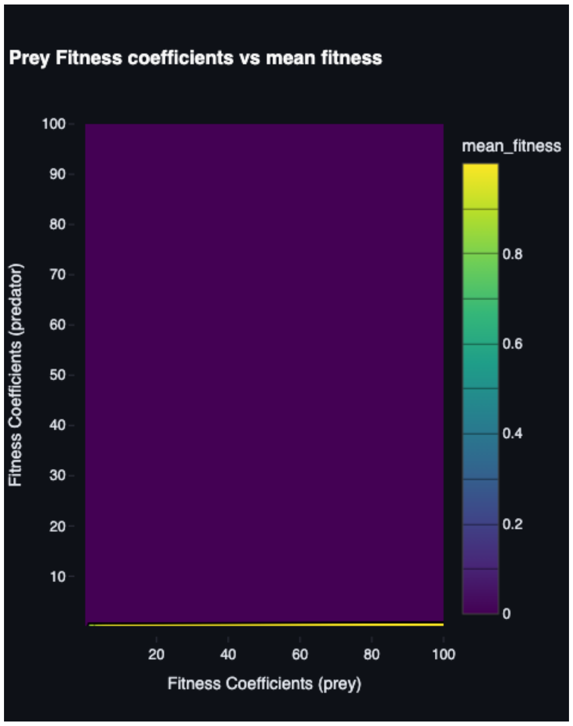


(a) Example 1

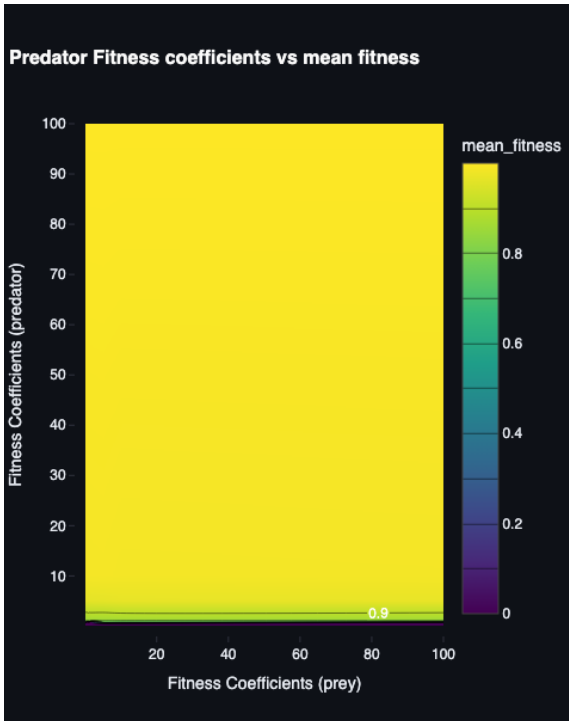


(b) Example 2

Figure 1: Lotka-Volterra like dynamics between prey and predator populations

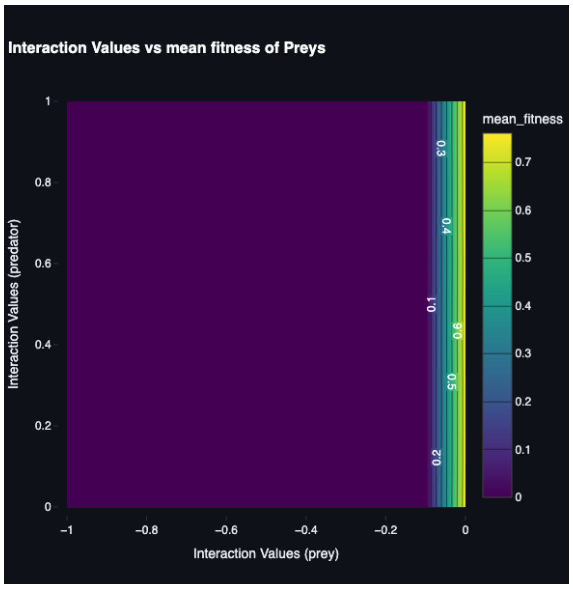


(a) Prey mean fitness

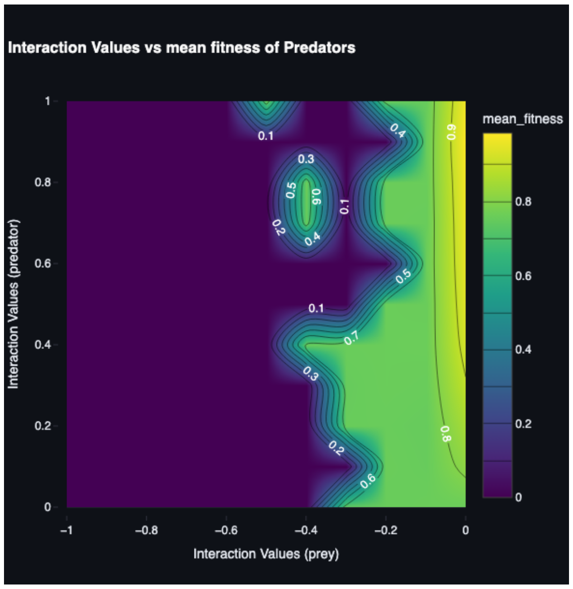


(b) Predator mean fitness

Figure 2: Contour plots of mean fitnesses over coefficient values

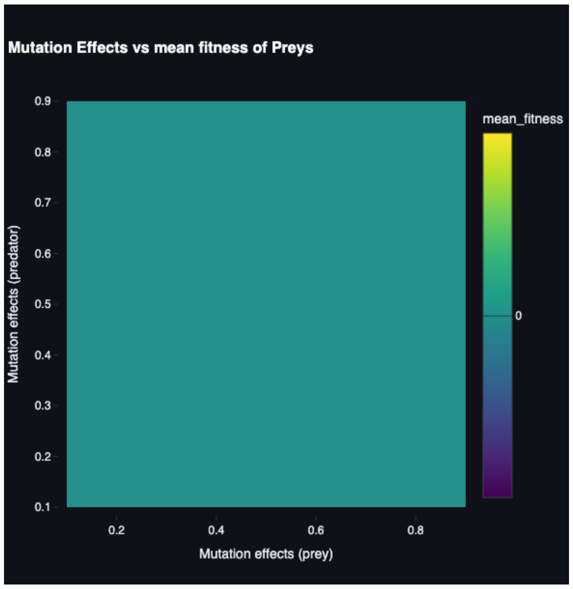


(a) Prey mean fitness

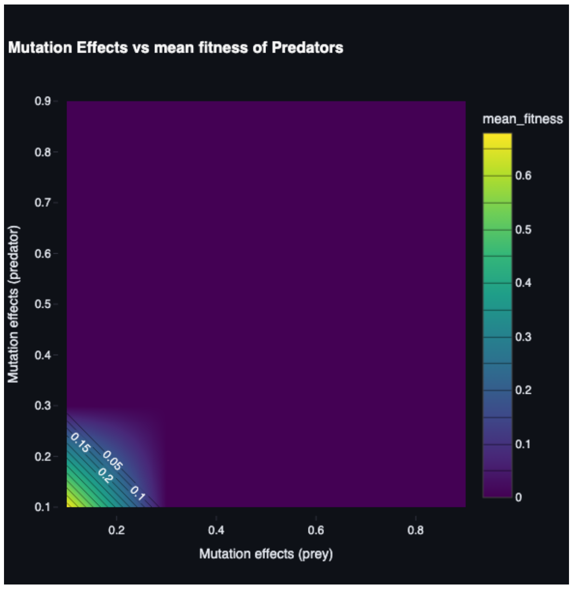


(b) Predator mean fitness

Figure 3: Contour plots of mean fitnesses over interaction values

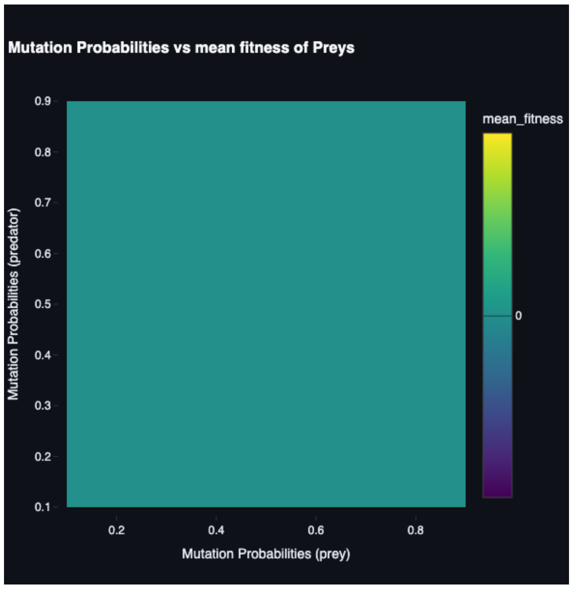


(a) Prey mean fitness

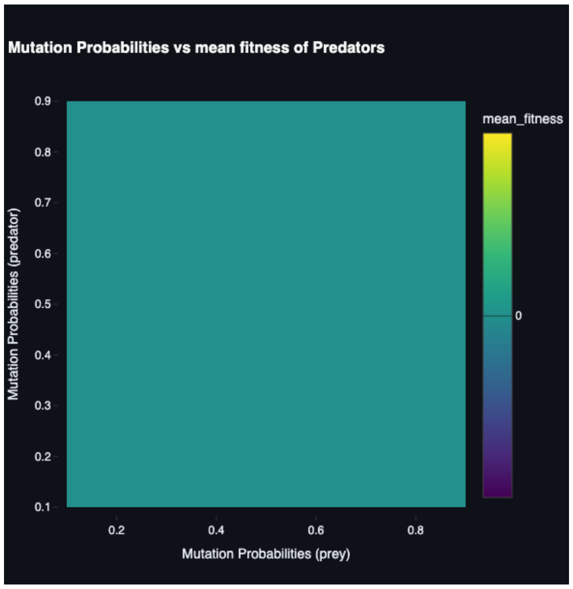


(b) Predator mean fitness

Figure 4: Contour plots of mean fitnesses over mutation effects



(a) Example 1



(b) Example 2

Figure 5: Contour plots of mean fitnesses over mutation probabilities